

SyncWare News

The journal for electronic
and other technical
applications of T/S
computers

Volume 2 Number 2

Nov.-Dec. '84

Sinclair QL Update

The latest word is that Sinclair will take orders as early as November, with deliveries as early as January (FCC permitting). Several improvements have been made to it since it was first introduced, mainly to correct the original discrepancies in the operating system, etc. At the current exchange rate, the QL may be quite a deal. We will see.

Memotech Revisited

All Timex Sinclair computer owners can expect to get an offer from Memotech soon. This offer is in the form of a trade in. They will reduce the price of their MTX512 computer a substantial amount, depending on what you have to trade in. The MTX512 has been upgraded considerably since it was first announced. We hope to have a review of this machine next issue.

Volume No.1 of SyncWare Soon

We are presently reconstructing and re-editing volume one of SyncWare News. We appreciate your patience. We are hoping to have Volume one out by the end of November. I'm sure that the original subscribers can comprehend the magnitude of this task. The original five issues amounted to 190 pages of 5"by 8.5", reduced type! If your mailing label has Vol. 1, in the first line, then you will receive this book when it is published.

Profile 2068

Contrary to popular belief, there is a Profile manual under construction. (I have seen the proofs with my own eyes.) There is however, one chapter to go before it is ready for the printer. The manual is comprised of not only instructions, but several modifications, program improvements and upgrades and will be a good tutorial on 2068 file handling in general.

FOR YOUR SUPPORT

This column announces any software or hardware that is new or otherwise untested by us. If you have something of interest, send us a description of your product. We advise readers to send a SASE, as a courtesy, to the individuals or companies, for further information.

ROBERT C. FISCHER, 221 Scoggins St., Summerville GA 30747, announces a gradebook program for teachers using the TS1000 or the TS2068. The program provides machine code data processing for multiple classes, and it provides for a flexible number of students and grades per class. \$15.95 for 1000 and \$19.95 for 2068.

UAS, PO Box 612, Haddonfield, NJ 08033, has a catalog listing of over 100 programs for BOTH computers. Included are games, utilities, business, finance and math programs. Prices from \$1.00 to \$19.95.

TOM LAFFIN, Bridge St., Box 133, Hillsboro, NH 03244, has previously owned ZX81 computers for \$25.00 each. Units each come with BASIC manual, ZX power supply, video switch and cable. Cassette cable not included.

KNIGHTED COMPUTERS, 707 Highland St., Fulton NY 13069, (315) 593-8219, has MULTI-DRAW for the 2068. It is a screen design program, giving joystick control of dual multi-function screen cursors which draw, print, define characters, change colors, flash, save/load or printout your own TV screen creations. The program works with the 2040 printer or a dot matrix printer and the Aerco interface.

RAMEX, 48945 Vandyke Rd., Utica, MI 48087, (313) 463-1795, has a catalog of software and hardware for the 2068. Programs include educational, scientific, accounting and games. Call or write for catalog. Ramex has a disc drive interface available around mid-November for \$200.00. It allows up to 4, 1 megabyte drives on your 2068.

BEN JOHNSON, Kaltek Calculator Technology, PO Box 7462, Rochester, NY 14615, has a few very nice kits for TS1000 computers. A Julian Clock kit is available for \$7.00 (board), \$26.00 for the full kit. A shift lock module kit is also available for \$17.46ppd. The RC-103 analog interface is available for \$18.00 and has a variety of applications, LOAD signal conditioner, ohmmeter, capacitance meter, light meter, tachometer, thermometer and others. An applications sheet comes with it.

FRED NACHBAUR, Syncware Co., 902 Hoover St., Nelson, BC V1L4X6, Canada, now has a RAM based Memotext available with Powell Hargrave's Super Data Save (SDS) relocated and incorporated for fast load and saves of text. Help screens and a return to BASIC are also included. Requires memory in the 8 to 16K block and the Memotech parallel I/F. Work with the RS232 I/F is in progress. This is THE word processor for the 81, 1000 and 1500. The entire contents of the last issue and the entire first volume were "Memotexted" on a ZX81. \$29.95, if you already have documentation, then the cost is \$24.95 (royalties, you know).

TOM BENT (that's me), 9016 Flicker Place, Columbia, MD 21045, (301) 730-7187, has an upgrade for your little ZX. Replace your 8K ROM with an improved eeprom. Full display file, SCROLL/CLS works properly, LPRINTs small numbers (.0001), DIM very large single strings (DIM A\$(47000)) in 64K and a few other nice changes are incorporated. \$20.00 for EPROM and circuit. (It fits under the hood.)

JIM HOUSTON ENTERPRISES, 414 West Elsmere Pl., San Antonio, TX 78212, has a shifter board which allows you to use the extra keys on a large keyboard. The bare board is \$17.00. It requires 40 resistors and diodes and 10 IC's. Assembled and tested, it's \$45.00, and w/edge connector \$55.00. Also, Jim has a direct video board and inverter for the TS1000 for \$15.00, he will install it for you for \$27.50ppd. He has some AERCO DISK system programs. For those of you interested, send him a SASE for details.

PAUL BINGHAM, Pleasantrees Programming, PO Box 7345, Mesa, AZ 85206, still has a few copies of Graphics A to Z for the 1000. The cost is \$21.00 ppd.

SUM-WARE, 810 Mammoth Rd., Alden, NY 14004, has the 2068 PINBALL program on the original cartridge. Supplies are limited. \$29.95 plus \$2.00 shipping.

RAY KINGSLEY, PO Box 8032, Santa Fe, NM 87504, now has HOT Z AROS. This program will run TRANSPARENT, and allow you to step through 3 64K banks of memory (now under development), read, write and execute code anywhere memory is available. Also, Sinware has available a 2764 eeprom to replace the bug infested EXROM. This is a monumental and much needed improvement to the underdeveloped 2068. Prices- HOT Z 2068 V1.8 and 1.9 on tape, allows access to any part of RAM, \$24.95, ROM annotation, an excellent source of 2068 info, \$15.00, Two 2764 eeproms burned w/HZ AROS \$39.95. (full bank switching). HZ AROS on cartridge board, \$59.95, 2764 EXROM replacement, \$16.00ppd. All other orders add \$2.00 each item for p/h.

GIBSON Data Systems

Announces

A full powered business program for the expanded T/S 1000, 1500, and new 2068:

The Bookkeeper

GIBSON Data Systems

9 Orchard Drive Durham, NH 03824

all prices include U.S. shipping

MASTERCARD

- up to 900 journal entries
- carry forward
- 99 user definable account names
- check register maintenance
- print out for:
 - journal with comments
 - ledger by account, date or all
 - chart of accounts
 - income statements
 - balance sheets
- item search
- easy corrections and changes
- for 32k or 64k with TS 2040 printer

Send \$25.00 for: Comprehensive manual
Cassette

FORUM

First, I would like to give the address of QZX, the Journal for Amateur Radio and Sinclair Computers. We neglected to give this important information in issue 1/5. Contact them at:
2025 O'Donnell Drive, Las Cruces, NM 88001
They have been around for three years now for those of you who are ham radio buffs.

Tony Gomez, 213 Los Feliz, Thousand Oaks, CA, (805) 497 9851, has a board that converts the Compusa disk interface to drive the DEC disc system.

There has been a ton of response to our first issue of volume two. Thank you. With round two of the questionnaires in, Product Reviews are weighting quite heavily towards the number one choice that people read about. I draw two conclusions from this. One, you want to know what is available and worthwhile, and two, you have been either burned by bad software or baffled by it. We need reviewers. We have received some response from our last plea, and we have several reviews "in the pot." We will have your fair share of reviews next time.

We have received numerous letters (thank you), and we will be printing what we have room for as time goes on.

Dear Editor---I must write and tell you that Basil's article has given me some real satisfaction. It was the motivation to renew my interest in Machine Code, that this article brought me...After struggling with Toni Baker's "Mastering Machine Code on your ZX81", I eventually realized that in a statement like DIM (A\$(22,32), that A\$(0) has no meaning. Anyway, I look forward to Basil's future articles.

Roy E. Brann, South Pasadena, CA 91030

First, A\$(0) is a throw back from the ZX80 4K ROM. Toni wrote that book quite some time ago. Basil replies, "Thanks for your letter of encouragement. I too struggled with that book. It tried to do too much in too little time. I may lean too far the other way. I am trying to cover every agonizing step, although I don't want you to lose interest. This text was originally a book, in which the reader could pick his own pace. This serialized rendition may go slow. Bear with me. I will try to include a nice little utility in each 'chapter.'"

Dear Editor---Please let me know who can fix my ZX81 if it breaks and tell me where I can get spare parts for my machine just in case.

Kingsley Langenberg, Waukegan, IL 60087

Parts? ZX81's have parts? The only part you can't get easily, is the ULA. It is available from Timex Computer Corp., Little Rock, Ark., for \$12.00. Most other parts can be bought from JDR, Dokay, Radio Shack, etc. (Check the back of a Byte magazine.) Look around at local flea markets. I have purchased brand new TS1000's in boxes for \$.99!! The box is worth that much. Don't hesitate to open the back and "play around" with it. It is a great machine to "LEARN" on. For problems, start with the power supply. Is it warm? NO? Replace it (9VDC 1 amp). Leave the machine on for 15 minutes. Is it warm over

the 2, 3 keys? NO? Replace the 5 volt (7805) regulator (Radio Shack). Do you have a display? NO? You have a problem. Crash with 16K ram hooked up? Right away? possible bad ram pack. (I recently bought a 16K ram for \$9.95) After 10 minutes? Maybe a wobble problem, weak power supply, static electricity problem or dirty edge connector. Check for VERY hot power supply, TV with a lot of static from the screen, etc. Won't LOAD or SAVE? Possible noisy RAM pack, check connectors and only use ONE connector at a time. Also, check the diodes near the LD/SV jacks. Keyboard row won't work? Check and firm up the ribbon connector. Check the diodes near the ribbon connector. Still have a problem? Try swapping chips. The SGS 180 may be weak. Still not running? You need some help. By the way, what IS the problem?

Is there anyone out there who would repair these little black boxes that we all hold so near and dear? Let us Know.

DECIMAL TO HEX AND BACK

Tom Woods

Converting from hexadecimal to decimal number systems has been the bane of many an aspiring machine code programmer. This listing does the converting for you. When you RUN the program, a prompt asks you for a hex number. You can either type one in to get the decimal equivalent, or press ENTER to flip to the decimal to hex mode, where you can type in a base 10 number to get the hex form.

The program can be used as a foundation for a much more elaborate monitor. Subroutines at lines 1000 and 2000 convert the number held in A\$ from hex to decimal or decimal to hex respectively. The program returns from the subroutines with the hex number in A\$ and the decimal number in C. An error flag (called ERR) is set to 1 if a faulty input is received.

```
10 LET ERR=0: DEF FN A(H$)=CODE  
E H$-48-7*(H$>"9"): LET H$="0123  
456789ABCDEF"  
20 INPUT "HEX into DEC?": A$:  
IF A$="" THEN GO TO 100  
30 GO SUB 1000: REM hex to dec  
35 IF ERR=1 THEN PRINT "INVALI  
D NUMBER": GO TO 1  
40 PRINT A$;" hex=";C;" dec."  
50 GO TO 20  
100 INPUT "DEC into HEX?": A$: I  
F A$="" THEN GO TO 10  
110 GO SUB 2000: REM dec to hex  
120 GO TO 35  
1000 LET C=0: FOR X=1 TO LEN A$:  
IF CODE A$(X)>48 AND CODE A$(X)  
<=70 THEN LET C=C+FN A(A$(LEN A  
$-X+1))*16+(X-1): NEXT X: RETURN  
2000 LET ERR=1: RETURN  
2010 FOR X=1 TO LEN A$: IF CODE  
A$(X)>47 AND CODE A$(X)<58 THEN  
NEXT X: GO TO 2010  
2005 GO TO 1010  
2010 LET C=VAL A$: IF C>65535 TH  
EN GO TO 1010  
2015 RANDOMIZE C: IF C=0 THEN RE  
TURN  
2020 LET P=23671: LET A$=""  
2030 LET N=PEEK P: LET N$=H$(N/1  
6+.5): LET N=N-INT (N/16)*16+.5:  
LET N$=N$+H$(N): LET A$=A$+N$:  
IF P=23670 THEN RETURN  
2040 LET P=P-1: GO TO 2030
```

ON LOADING

Ed Shaughnessy
151 Daniel Low 3E
Staten Island, N.Y. 10301

The program presented here creates a modified LOAD command. It moves the routines needed for the LOAD from ROM into RAM and adjusts them to work there. The routines are changed so that when the LOAD completes, machine language code from the 1 REM statement is executed. By changing statement 1, you can completely control what occurs after the LOAD.

Background Information

When a program is SAVED, the ZX/TS computer begins by writing a lead-in of five seconds of silence. Then it writes the program name ("header"), followed by the four parts of the program: the system variables, the BASIC program, the display file (i.e. the television picture,) and the BASIC program variables. When reloading the program, the LOAD command reads the tape until it finds the header with the program name, and then loads the four parts of the program into the computer.

Two routines are used by the LOAD command. These are the load routine (locations 0340-03A1h of ROM) and the terminate routine (01FC-0206.) Every time the load routine reads a character from tape and writes it into the computer, it calls the terminate routine which checks to see if the end of the program has been reached. When it finally gets to the end, the terminate routine does not return control to the load routine; instead, the code immediately after the routine is executed. Normally this is the display routine.

The program shown here moves the terminate routine and the load routine, as well as everything between them, to the top 1K of RAM. The address of the assembler statements (the CALLs and JPs) are adjusted to work in the new location. Then, before calling the LOAD command in RAM, the program moves the machine-code in the 1 REM statement into the area of RAM immediately after the terminate routine. This code will execute **after** the LOAD finishes. Two possible versions of this modified LOAD command will demonstrate what can be accomplished by carefully selecting the code that is stored in the 1 REM statement.

SAVE the UnSAVEable

Suppose you purchase a self-running program and you wish to make a back-up copy. This is difficult or impossible to do the usual way because as soon as you load the program, it executes. Version 1 of the modified LOAD command solves this problem. We have seen that when the LOAD command in RAM completes, the code from 1 REM executes. In version 1 of the modified LOAD, this code calls the keyboard scanning routine until the "S" key is pressed. Then the address of the name of the program to be saved is set to a location in ROM that happens to contain an inverse "X." Finally, a jump is made directly to the SAVE routine in ROM.

Type in the program of version 1. Type RUN, then ENTER. The screen will become white while the code is being moved from ROM to RAM. When the modified LOAD command begins to execute, the loading pattern will appear on the screen. Start playing the tape that you wish to back up. The loading pattern will continue until the load completes, and then the screen will turn gray. Place a blank tape in the recorder, switch the cassette lead to the MIC sockets, and press RECORD. Now press the "S" key to begin the SAVE. After five seconds, the saving pattern will begin. At the end of the save, the screen will go white with the report C/190.

You now have a backup copy just like the original. To LOAD the copy you have just made, use the command LOAD "X". The program will run as soon as it is loaded.

```
1 REM E1CD8B02EB21FDFBA7ED522
0F421DB04C3FC020000000000000
10 REM SAVE A SELF-RUN PROGRAM
20 FAST
30 LET A=124
40 FOR I=508 TO 929
50 POKE I+A*256,PEEK I
60 NEXT I
70 LET B=A*256+834
80 POKE B,55
90 POKE B+7,A+3
100 POKE B+32,6
110 POKE B+33,A+2
120 POKE B+38,A+3
130 POKE B+60,A+3
140 POKE B+64,A+1
150 FOR J=1 TO 25
160 LET C=16512+J*2
170 POKE J+A*256+517,(PEEK C-28
)+16+PEEK (C+1)-28
180 NEXT J
190 RAND USR B
```

LOAD the UnLOADable

Suppose last night you keyed in a long program and saved it on tape. Now you try to load it. The loading pattern appears for half a minute or more and suddenly the screen goes white for a few seconds and an inverse K appears at the bottom. When you try to LIST the program, you receive a report 0/0, indication that nothing was loaded. I'm sure all SyncWare News readers have encountered this frustration. There is a bad spot on the tape, resulting in a drop-out. When the LOAD command reaches a silent spot on the tape, it jumps to the NEW command which deletes whatever has already been loaded. Since you have a program listing on paper, you would like to recover the program up to the bad spot on the tape, so at least you won't have to re-enter the entire program. Version 2 of the modified LOAD routine will load a tape up to the drop-out.

When the modified LOAD command reads a silent spot on a tape, it jumps to the area after the terminate routine which holds the code from our 1 REM statement. In version 2, this code determines how far the load has gotten. If the bad spot occurred after the system variables and the BASIC program were loaded, whatever was loaded from the display file or program variables area will be disregarded.

If the drop-out occurred while the BASIC program was being loaded, then the end of the last complete BASIC line is found so you won't get a program that is unlistable because part of the last statement is missing. Finally, a jump is made to a routine in ROM that will create an empty display file and an empty program variables area.

Type in version 2 of the program from the listing. After saving it for future use, enter RUN; the screen will turn white while the routines are moved into RAM. When the loading pattern appears, start the tape recorder in PLAY with the bad tape. The loading pattern will continue until the bad spot on the tape is reached. Then the screen will go blank for a few seconds and then display report 0/0. You can now LIST. If the program is not complete, type in the missing statements.

```

1 REM EB2A0C4037ED522A0C40380
AEB0100003E76EDB92323C30304
10 REM LOAD PART OF A BAD TAPE
20 FAST
30 LET A=124
40 FOR I=508 TO 929
50 POKE I+A*256,PEEK I
60 NEXT I
70 LET B=A*256+834
80 POKE B,55
90 POKE B+7,A+3
100 POKE B+32,6
110 POKE B+33,A+2
120 POKE B+38,A+3
130 POKE B+50,A+3
140 POKE B+54,A+1
150 FOR J=1 TO 25
160 LET C=16512+J*2
170 POKE J+A*256+517,(PEEK C-28
)*16+PEEK (C+1)-28
180 NEXT J
190 RAND USR B

```

Final Words

The value in A in statement 30 determines where in RAM the routines are moved to. Readers with RAM in the 8-16K block can change the value of A so that the LOAD command is moved to an area safely outside the usual 16K RAM area. Then even the largest programs being loaded in will not overlay the routines in RAM and cause a crash.

Readers with disassemblers that reside in the top part of RAM (e.g. Bug-Byte's ZXDB) may also set A to a different value to avoid a conflict. Make the following changes:

30 LET A=96
Delete line 190

Run the program to move the routines to a location in RAM that won't conflict with the disassembler. Use the disassembler to view the assembly code in locations 61FC-621E and 6242-62A2.

For a detailed explanation of the LOAD command, see "The Explorer's Guide to the ZX81 and Timex Sinclair 1000" by Mike Lord. Another book that is valuable for anyone wanting to understand ROM routines is "The Complete Timex TS1000/ Sinclair ZX81 ROM Disassembly" by Ian Logan and Frank O'Hara. I hope that by using the program shown here, readers will discover other useful changes to the LOAD command.

WINKY BOARD Cassette-Computer Interface

- * Solves your LOADING problems.
- * Duplicates ANY TS/ZX cassette
- * User friendly. Simply plugs into cassette-player & computer jacks.
- * Makes UPLOADER use easy on TS2068.

WINKY BOARD 2000 for TS2068, ZX Spectrum, TS1000/1500, ZX80/81 Price \$22.95 assembled/tested, shipping incl. to U.S./Can.

WINKY BOARD2 for TS1000/1500, ZX80/81
Price \$18.95 assembled/tested; \$15. kit

SPEECH RECOGNITION SYSTEM

- * Train your computer to obey voice commands
- * Recognizes up to 8 spoken words
- * Includes speech amplifier unit, cassette program, documentation

SAS2000 for TS2068

SAS1000 for TS1000/1500, ZX81

\$32.95 assembled/tested; \$27.95 kit

COMPCOOLER Power supply-Computer Interface solves most overheating problems for TS1000/1500, ZX80/81 \$6.95

HI RES GRAPHICS PLANS - TS1000, ZX81 interface \$5.

Utilities on cassette for TS1000/15000, ZX81

KEY - unlock, merge, renumber \$9

HI RES PRINTER GRAPHICS \$9

ZXLAS Fastload (not for 1500) \$10.50

Prices above include shipping to U.S./Can.

NEW LIFE FOR YOUR TS2068?

YES! with

ROMSWITCH

Lets you run Spectrum programs on your TS2068!

- * EASY INSTALLATION. No soldering, no drilling.
- * Just plug inside your TS2068 freeing edge connector & cartridge port for other uses.
- * External Stick-on switch selects Spectrum or TS2068 ROM
- * Thousands of good British programs available, many from U.S. dealers
- * Runs more programs than Emulator/Chameleon.

Price \$54.95 assembled/tested, U.S. shipping incl.
Canadians add \$2., overseas add \$5. shipping

Information sheet, list of U.S./Can. retailers of Spectrum cassettes, & our complete catalog free.

RUSSELL ELECTRONICS



RD 1 * Box 539 * Centre Hall, PA 16828

814-364-1325 MasterCard/Visa 10am-8pm Check/MO

TRANSLATIONS

Memotext in RAM

by Fred Nachbaur
902 Hoover St.
Nelson, BC V1L 4X6
Canada

Here is how to make a copy of the "Memotext" word-processor, which you can save to tape and re-load later into RAM or NVM (non-volatile memory) in the 8-16K range. This will be useful if you, as I, have a second machine with Memotech CIF (Centronics printer interface) and wish to run Memotext without "tearing down" your main machine (or buying a second module, even if you can find one; Memotech no longer supplies them.) Or, the Memotext module may increase your ZX81/TS1000's power demand to the point where you get crashes after extended operation on a hot day; if you could substitute an NVM board such as the Hunter board with its lower power consumption, you wouldn't have to build or buy a larger power supply to accommodate Memotext. Lastly, TS1500 users will find this very useful, as the module runs into the cassette plugs, whereas the Hunter board should just barely fit.

Before we get into the step-by-step instructions for making a RAM-based version of Memotext, I should mention that this will require NO hardware modifications to your machine, and no additional software is needed to do the job. Although a "REM generator" is useful for the first step, it is not vital as I'll describe a (relatively) quick and easy way of doing this "by hand." The hardware required is a ZX81/TS1000 with at least 16K user RAM. Also you'll need the CIF and (of course) Memotext modules. If you have the SIF (RS232 serial interface) the procedure should work equally well, though we haven't actually tried it. The machine on which you run the final result should have, again, 16K RAM or higher, CIF (or SIF,) and RAM or NVM in the full 8-16K range (such as is afforded by a 64K pack or a Hunter board.)

Lest you're worried about "conflicts" due to having the CIF module and RAM in the same area, fear not. We'll duplicate the CIF (or SIF) program in the RAM, so that when this area (10-10.5K for CIF, possibly more for SIF) is addressed, both units return the same output on the data lines; result - both units "give the same answer" and no conflict can occur.

BACKGAMMON

Looking for a real challenge? Tired of slow response time? Practicing for a tournament? If so, you'll want to try our Backgammon game. Features: full-screen graphics, break disabled, 100% machine code, clean loading, smart play, top rating from TSUB review, multi-game scoring, cube doubling, simple operation, documentation included. Requires ZX81 or TS1000 w/16K RAM. Available on cassette for only \$10. Purchase before 2/1/85 and receive our Z80 dis-assembler machine code monitor at no additional charge! Write for our FREE catalog listing 100 programs for TS 1000 and 2068. Please include exp. dt & phone # on MasterCard/VISA orders.



BIOCAL
SOFTWARE, INC.
167-C Wilson Street
Petaluma, CA. 94952

An additional bonus is that the unused area from 10.5-12K will be available for other routines, such as fast load/save programs, transfer routines, or other utilities. While we're at it, we'll also patch in a "QIT" function to allow return to BASIC from Memotext without having to power down or hit a reset switch. We'll also add an alternate initialization step to preserve your current RAMTOP setting. Memotext, as supplied, uses up all available memory regardless of the initial RAMTOP setting, which can be a pain if you're keeping other programs in "protected memory." In an earlier issue I reported that you could keep protected programs in high memory, which remain there until you hit your RESET button to escape Memotext. This is true only if you're careful not to go beyond 16K of text or data files; with the modified entry point, you can type away to your heart's content until you get an "out of space" message. On entering QIT, you leave Memotext with your RAMTOP setting intact.

But enough introductory ramblings, already. Here's how you go about performing this feat of magic.

1. Generate a REM ...

Power up your ZX81/TS1000 with Memotext, CIF, and at least 16K RAM. Disable the 8-16K region if you're using a 64K pack. Now generate a line 1 REM statement at least 8204 bytes long (8192 bytes for Memotext and CIF, 12 bytes for the transfer routine that will upload then later download the program.) If you have a REM generator utility, use it and proceed to step 2. Otherwise, go on to step 1A.

1a. ...the Easy Way

If you had to type in an 8K REM statement a character at a time, it would probably be enough to make you give up computing for good. Fortunately, you can use a couple tricks to help matters immeasurably. First, go into FAST mode and type in a line 1 statement like this:

```
1 PRINT 0+0+0+0+0....+0
```

You will need a 1 PRINT 0 followed by 113 repetitions of "+0" - for a total of 114 zeroes and 113 pluses. Each "0" actually takes up seven bytes, so you've actually entered a statement 911 bytes long. (See "VDAQ" in SWN 1:5 for an explanation why this works.) Note that the keyword must be a PRINT, not a REM. Check that you have the right length; PRINT PEEK 18342 should return 118 (end of line "enter" character.) Now make eight more identical lines, using the edit function (shift 1) to bring down the line, change the line number to 2, and enter; keep doing this until you have nine lines, numbered consecutively 1-9. Now bring line 9 back down, and delete five repetitions of "+0" - and re-enter this slightly shorter version. At this point, check your work with PRINT PEEK 24721; you should again get 118.

Now enter the following "immediate" commands
(no line number:)

```
POKE 16511,17
POKE 16512,32
POKE 16513,234
```

The first two POKES tell the computer that the line 1 is actually 8209 bytes long (including the PRINT and the "enter" at the end)- i.e. all the way to the end of line 9. The third POKE changes the first PRINT into a REM - you now have an 8207 byte REM statement. When you LIST 1, everything still looks the same, but you can verify that all nine lines have indeed been "swallowed up" by trying to LIST (or delete) lines 2, 3, etc. and you'll see that these lines no longer exist, as such.

2. Put Memotext Into the REM

Now enter the BASIC lines of Listing 1. When you've checked them over for accuracy, hook up your tape recorder, put it in RECORD, and RUN the program. It will save to tape (in case you make a boo-boo later) and then prompt you to input the 12-byte transfer routine to load Memotext and CIF into the REM statement. Enter the values from Table 1; when all 12 have been entered, check that they're all correct (if not, answer "N" on "ALL OK?" prompt, and do it over.)

LISTING 1.

```
10 SLOW
20 SAVE "MEMOTEX"
30 CLS
40 PRINT "INPUT TRANSFER ROUTI
NE"
50 FOR A=24706 TO 24717
60 INPUT B
70 POKE A,B
80 PRINT A,PEEK A
90 NEXT A
100 PRINT "ALL OK?"
110 IF INKEY$="Y" THEN GOTO 140
120 IF INKEY$="N" THEN GOTO 30
130 GOTO 110
140 CLS
150 PRINT "TURN MEMOTEXT ON"
160 PAUSE 300
170 RAND USR 24706
180 PRINT AT 0,15;"FF","THEN BR
EAK"
190 PAUSE 4E4
```

Assuming that you did it right, press "Y." You now have five seconds during which to flip up the switch on Memotext. Don't touch the keyboard while the Memotext module is on, or you might unwillingly enter the Memotext program. At the end of the five-second PAUSE, the transfer routine you laid in will almost instantly transfer the entire 8-16K region of memory (Memotext and CIF) into your line 1 REM. When prompted, flip Memotext OFF and then hit the BREAK key. If you LIST line 1, you will see that typical "machine-code-ese" has replaced all those PRINT 0+0+0's.

At this point, once again SAVE to tape, this time from the immediate mode.

3. Making a few Changes

After SAVEing to help protect against Murphy's Law, make the following POKES:

```
POKE 24710,0
POKE 24711,32
POKE 24713,130
POKE 24714,64
```

These POKES reverse the direction of the transfer routine, so you can later boot the whole works back down into the 8-16K RAM from the REM statement.

```
POKE 20951,54
POKE 20952,46
POKE 20953,57
POKE 20954,195
POKE 20955,3
```

These POKES install the "QIT" function in the program's function table. You can install your own additional functions starting at 20956 (later to be located at 12634,) e.g. to call other utilities from within Memotext. Five bytes are required for each function; the first three are the character codes of the function (Q, I, T in our case) and the last two are the address which is called by that function (low-high order - the BASIC "NEW" command in the case of the QIT command above.) There is room for up to three more commands after the QIT.

```
POKE 17245,42
POKE 17246,4
POKE 17247,64
POKE 17248,195
POKE 17249,90
POKE 17250,54
```

These POKES set up a alternate entry point to the program. This entry point makes the program use the existing RAMTOP setting in initializing its stack, leaving the rest of "protected" memory alone.

Now delete BASIC lines 30 through 190, (leave in lines 10 and 20, and of course the line 1 REM) and again SAVE to tape using RUN (just in case of a screw-up.)

TABLE 1

ADDRESS	VALUE	ADDRESS	VALUE
24706	1	24712	33
24707	0	24713	0
24708	32	24714	32
24709	17	24715	237
24710	130	24716	176
24711	64	24717	201

4. Enter the Basic 'Loader'

Now enter lines 30 through 130 of Listing 2. Use tokens for words like AND, TO, RETURN, etc. in the PRINT statements if you wish to save a few seconds loading time. When you've checked for errors and corrected any you find, the RAM-based Memotext program is complete. SAVE to tape by entering RUN.

```

10 SLOW
20 SAVE "MEMOTEX"
30 POKE 8192,0
35 POKE 16383,0
40 POKE 8192,201
45 POKE 16383,201
50 IF PEEK 8192<>201 OR PEEK 1
6383<>201 THEN GOTO 100
60 RAND USR 24706
70 PRINT TAB 3;"*****MEMOTEXT
STORED*****";"FLIP WRITE-PROTE
CT IF AVAILABLE.";"ANY KEY TO "
"NEW " "LOADER, THEN RAND USR 1
3824 TO ENTER MEMOTEXT(MUST BE I
N " "SLOW " "MODE.)";"TO EXIT
MEMOTEXT AND RETURN TO BASIC,
ENTER "QIT" IN RESPONSE TO "
FUNCTION?" PROMPT."
75 PRINT " " TO RESTART MEMOTE
XT;" " RAND USR 13824 OR 13890
TO USE "ALL AVAILABLE RAM"
OR " " RAND USR 8923 TO PRESERV
E RAMTOP"
80 IF INKEY$="" THEN GOTO 80
90 NEW
100 PRINT "*****NO RAM IN 8-16K
REGION*****";"ENABLE NVM OR 8-
16K RAM, THEN "PRESS ANY KEY TO
CONTINUE."
110 IF INKEY$="" THEN GOTO 110
120 CLS
130 GOTO 30

```

5. Testing the Program

After SAVEing, power down your computer and remove the Memotext module and install your NVM board or engage the 8-16K region of your 64K pack. Leave the CIF (and your printer) on-line, and power back up again. LOAD the final version of the program using LOAD "MEMOTEXT" - when loaded, the program checks the first and last bytes of the 8-16K range to verify that it's there and can be written to; if you get the "No RAM in 8-16K region" message, you've probably just forgotten to enable it; do so now, and press any key. You should get the "Memotext Stored" report, indicating that the program has been successfully booted into low memory. You're now ready to run Memotext.

Note that you have two (or rather three) options for entering the Memotext program itself. The first, accessed by RAND USR 13824 or RAND USR 13890 (use whichever you find easier to remember) causes the program to initialize as before; it checks to see if you have 16, 32, or 48K of user RAM, automatically sets RAMTOP and its stack accordingly, and proceeds into the program. The other entry point, accessed with RAND USR 8923, leaves RAMTOP alone and initializes to this point only. To escape Memotext after saving your files to tape, simply enter "QIT" from the "Main Program Loop" ("FUNCTION?" prompt.) Whichever entry you use, the program should operate exactly as detailed in the manual.

To make back-up copies of the tape, simple LOAD it, then press BREAK when the screen report appears; put in a blank tape, start the machine in RECORD, and enter RUN.

Final Comments

In case you've wondered how the Memotext module jumps into the program simply by pressing a key after flipping it ON, here's how it works. The module's hardware actually changes four ROM locations in the "warm boot" portion of the operating system. It changes the command at 0416h from CALL 0F23 (FAST) to CALL 0207 ("Mode check,") and changes the command at 0419h from CALL 0A2A (CLS) to CALL 3600 (Memotext entry.) This part of the ROM is executed when you enter a command in the immediate mode - hence, the "auto-start" feature. In the RAM-based version, of course, this doesn't happen, since we haven't added the hardware to "jimmy" the ROM.

As indicated in the "title screen," your USR call to Memotext must be made from SLOW mode. As near as I can tell, the program still works in FAST mode, though it doesn't do you any good since you can't see the screen display.

It is quite probable that a similar procedure will work to make RAM-based versions of the other Memotech program modules (Memocalc and Memotext assembler) as well as the Timex "Command Cartridges." I'd be interested in hearing from anyone experimenting with this, and the results. Quite probably, in the case of the Memotech stuff, the same ROM changes are made; you can check this out by using HotZ (or a BASIC loop) to make a copy of the ROM in high memory with the module on, then quitting HotZ, and writing a simple BASIC loop to compare the actual ROM with the copy in hi-mem.

I would also be interested in hearing from anyone experimenting with the RAM-based Memotext using the JLO video upgrade. I haven't finished building up my "Super ZX" with the JLO TMS 9918 upgrade, so haven't been able to mess with it myself. It could be that the increased speed would make the auto-repeating key feature too fast; another challenge would be to find and change the timing loop that controls this.

This entire article was, of course, written using the the RAM-based version of Memotext. So if it doesn't work for you, chances are something went wrong along the way - go back to the beginning and try again.

**Support Our
Advertisers
Because Our
Advertisers
Support Us**

BASIL'S COMPENDIUM

Basil Wentworth
1413 Elliston Drive
Bloomington, IN 47401

PEEK, POKE & USR

This chapter will review the operation of the ZX81 memory, it will discuss PEEK and POKE and it will introduce the USR command.

First, let's take a look at another utility program on a "play-by-ear" basis, just to keep up the interest. This one is the classic line renumber program. (Fred Nachbaur would call it a half-program, since it does not make appropriate adjustments in GOTO and GOSUB destinations, but it's still pretty impressive--especially when you remember that it is only 34 bytes long.)

Now, enter the "rough draft" 1 REM line shown in Figure 2-1, and proof read it via the rudimentary loader program of Figure 2-2. (Keep in mind that it will not run until all the periods (.) have been poked with the data in table 2-4. The first column is the address+ 16500 and the second column is the data to be poked.-ed.) You'll find the loader a little different from the one in the last issue, in that only one column of figures is displayed now. There's nothing sinister in this change--it's just that the new form is easier to work with when it has to handle a number of different programs. The number of bytes has also been changed, of course, to fit the length of the new program.

After you've proofread the line (Figure 2-3), POKE in the vacancies, as shown in Figure 2-4, and proofread the final version as per Figure 2-5. Now you'll find that the command, RAND USR 16514, changes all your program lines into a neat sequence. It probably louses up your whole program if it has any GOSUBs or GOTOs in it though

If you want a different line number sequence, you can change the interval between line numbers by POKEing the desired number into 16539. You can change the number of the second line by POKEing your desires to 16515. (All line numbers must be less than 256, of course.) Line number one will remain unchanged, no matter what you do with the program. Now, back to our general discussion.

FIGURE 2-1. 1 REM FOR RENUMBER
-FIRST DRAFT-

```
1 REM " 5 RANDY.W74.7 COS Y5  
7. FAST 5 LPRINT .RND
```

HOW IT LOOKS

```
1 REM (GR 1) (GR 5) (SPACE) 5  
(GR 0) RND Y . W 7 4 . 7 COS  
Y 5 7 . FAST 5 (GR 5) (SPACE)  
(GR 0) . . LPRINT . (GR  
H) RND
```

HOW IT IS DONE

FIGURE 2-2. THE RUDIMENTARY LOADER

```
10 REM PROOFREADING THE PROGRA  
M  
20 FOR F=16514 TO 16547  
30 PRINT F;TAB 6;PEEK F;TAB 10  
CHR$ PEEK F  
40 NEXT F  
50 STOP  
100 REM POKEING THE VACANCIES  
110 INPUT A  
120 PRINT A;TAB 6;  
130 LET A=A+16500  
140 INPUT B  
150 POKE A,B  
160 PRINT PEEK A  
170 GOTO 110
```

ZX/TS Memory

You will remember that your computer stores information in the form of "bytes," each of which is an integer between 0 and 255, inclusive. Every time you press a key (except for SHIFT, DELETE, and such), that keystroke is translated into the appropriate byte, and stored in memory. (Sometimes the computer volunteers a few extra bytes that you didn't ask for. We'll call them "housekeeping" bytes, and discuss them later on.)

To see what these bytes look like, enter and RUN the mini-program shown in Figure 2-6. Note that 16509 is the address of the first byte of the program.

16509 is the address of the first program byte. That's a number worth remembering!

What you see on the screen is the contents of the first 50 bytes in the program area of the RAM. With a few exceptions, these bytes correspond to the keystrokes that you entered. These exceptions, the housekeeping bytes that we mentioned above, are very important to the writer of machine code. Don't worry, we'll get around to them before too long.

The PEEK command lets you look at the contents of any memory location. POKE, on the other hand, lets you change the contents of any RAM location you choose, to any number you specify (0 to 255 inclusive).

To watch POKE at work, leave the program of Figure 2-6 in the computer, and enter the command:

POKE 16518,35

You'll find that line ten has been changed to read:

10 FOR M=16709 TO 16558

To see the significance of the bytes stored in memory, restore line 10 to its original form, and change line 20 to read:

20 PRINT CHR\$ PEEK M;

RUN the program. Do you see the words and characters that you recognize from the program? The apparently meaningless stuff represents those "housekeeping" bytes.

FIGURE 2-3. PROOFREADING THE PROVISIONAL 1 REM

16514	1	■	16531	00	■
16515	00	■	16532	100	■
16516	00	■	16533	70	■
16517	00	■	16534	7	■
16518	00	■	16535	7	■
16519	00	■	16536	7	■
16520	00	■	16537	0	■
16521	00	■	16538	0	■
16522	00	■	16539	0	■
16523	00	■	16540	0	■
16524	00	■	16541	0	■
16525	00	■	16542	0	■
16526	00	■	16543	0	■
16527	00	■	16544	0	■
16528	00	■	16545	0	■
16529	00	■	16546	0	■
16530	00	■	16547	0	■

ZX/TS Character Set

If your going to use the El Minimo Loader, described later on (which we have been using with our demonstration programs, without telling you about it), it will be useful for you to understand the computer's character set.

The entire set is printed in your instruction book. Alternatively, you can look at the character corresponding to any number N (always 0 to 255, inclusive) by using the command:

```
PRINT CHR$ N
```

You can print out the whole set by the program in Figure 2-7, if you prefer.

To understand the POKE exercise you just went through, enter the command:

```
PRINT CHR$ 35
```

The computer will return the symbol 7. Maybe you have deduced that 16518 is the address of the digit that you just changed from 5 to 7.

FIGURE 2-4. POKING THE VACANCIES

ADDRESS	VALUE
+16500	
21	117
22	250
23	240
24	112
25	113
26	00
27	7
28	195

The USR Command

The trigger for a machine code routine is the symbol USR (under the L key). USR must always be followed by a memory location telling the computer where to start executing the machine code. For the moment, let's represent that address by LL.

Whenever the computer senses this trigger, as in PRINT USR LL, RAND USR LL, LET A =USR LL or the like, it reacts like a Pavlovian dog, jumping into the memory at LL, and scooting through the program like a mad fiend, at a speed of tens of thousands of operations per second. If there are no instructions to the contrary, the sequence of operations will follow the program bytes in the order in which they occur.

FIGURE 2-5. PROOFREADING THE FINAL 1 REM

16514	1	■	LD BC,5
16515	00	■	
16516	00	■	
16517	00	■	
16518	129	■	LD HL,16513
16519	64	■	
16520	00	■	
16521	117	■	LD A,117
16522	00	■	
16523	00	■	INC A
16524	100	■	INC HL
16525	00	■	CP (HL)
16526	00	■	JR NZ,-4
16527	00	■	
16528	190	■	INC HL
16529	00	■	CP (HL)
16530	00	■	RET Z
16531	00	■	LD A,33
16532	190	■	CP (HL)
16533	240	■	RET M
16534	112	■	LD (HL),B
16535	00	■	INC HL
16536	113	■	LD (HL),C
16537	00	■	PUSH HL
16538	00	■	LD HL,10
16539	00	■	
16540	00	■	
16541	00	■	ADD HL,BC
16542	00	■	LD B,H
16543	00	■	LD C,L
16544	00	■	POP HL
16545	00	■	JP,16520
16546	00	■	
16547	00	■	

If the program (as per listing) encounters the RETURN instruction (address 16529 or 16533), the computer will RETURN to BASIC. If there are no valid RETURN addresses (the Stack Pointer, SP, will contain the address that the computer will RETURN to), well, you are on your own. Stand by to pull the plug. At this point, of course, you have lost everything in memory. Believe me, it's easier to include RETURN (extra, well placed RETURNS won't hurt you).

FIGURE 2-6. A PEEK AT THE MEMORY

```
200 FOR M=16509 TO 16558
210 PRINT PEEK M;" ";
220 NEXT M
```

If the trigger command is PRINT USR LL, the machine code routine will end by printing the value of a variable (register) known as BC. More precisely, it will print:

```
C + 256 * B
```

based on the values that the registers B and C have taken at the time the program RETURNS to BASIC. If the USR command does not include PRINT, the output may be in the form of some other action based on the value of BC. For instance, GOTO USR LL and LIST USR LL are both perfectly valid commands, as long as LL has a value that can be handled by the computer.

In all cases, the computer will enter the machine code at LL, execute the machine code until it meets a RETURN, and then try to execute the BASIC command, substituting the final value of BC for USR LL.

That's all for now, but there will be more.

FIGURE 2-7. THE ZX81/TS1000 CHARACTER SET

```
300 FOR M=0 TO 255
310 PRINT CHR$ M;" ";
320 NEXT M
```


REVIEW

Timex Sinclair Intermediate Advanced Guide

Oscar Marsh
PO Box 317
Manomet, MA 02345

Author--Jeff Mazur
Published by--Howard Sams, PO Box 7099,
Indianapolis, IN 46206
#22226, 232 pages, published 1984
\$9.95 plus \$2.00 shipping

This book was mentioned in an early SAMS offering, but due to the Timex pull-out it was not marketed until September. The demand was strong enough for it.

The book starts with a brief but thorough explanation of the insides of the 2068, its architecture and its electronics. A full explanation of digital mathematics follow.

Chapter 2 explains the binary number systems, fully charted and clearly understandable; I now know that I will probably never program in machine language!

Chapter 3 introduces the computer itself, what happens and why, at various points in its use. This chapter takes you from power up to power down, and gives you a chance to understand the goings-on inside that plastic enclosure.

Chapter four explains Sinclair usage of the Z80 CPU and how you can take advantage of this usage (do your homework first). Chapter five details the memory map; Chapter six, modems; Chapter seven, the sound generator and chapter eight entails the video display.

Part Two is devoted to various aspects of machine language and BASIC, and leads the reader, chapter by chapter, along the pathway to understanding the diverse operations conducted by machine language operations.

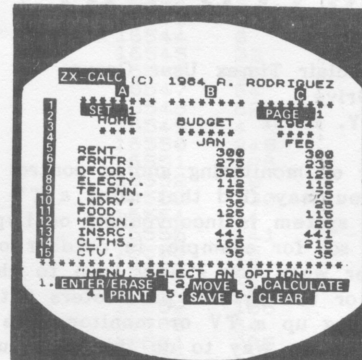
As a follow up to Fred Blechman's Beginners/Intermediate Guide, this book is exactly what you will need as your programming skills mature.

fig. 13-0 Two Toms at work at the esteemed editorial offices of the SyncWare News



POWERFUL AND INEXPENSIVE BUSINESS SOFTWARE FOR ZX81, T/S1000 and T/S1500 COMPUTERS

ZX-CALC



An electronic spreadsheet calculator is the fundamental basic tool for summarising, reporting and analyzing in matrix form any accounting, mathematical or scientific manipulation of numbers. ZX-Calc operates in 32-64K RAM and affords a maximum of 3360 characters / spreadsheet. The entire matrix consists of 15 columns (letters A-O) and 30 rows (numbers 1-30) with 8 characters / cell. Unlike other popular ESCs, ZX-Calc uses in calculations and within cells all 14 math functions on the ZX-81 / TS1000. It offers a unique *SUM function that totals one or more rows / columns simultaneously. Parenthesis can be used within equations. There is no fixed limit on how many equations may be entered. Formulas may be stored in all 420 cells of the spreadsheet. The display affords 15 rows / columns. Loading of data into more than one cell can occur across / down one or more row / column simultaneously. With vertical windowing you can arrange a set of columns in any order, or practice using fixed-variable-alignment display formats. The menu offers 6 options: enter / erase, move, calculate, print, save and clear the spreadsheet. Enter / erase allows the entering, deletion or data alignment within a cell through the use of a mobile cursor. With the move option you may move around the entire spreadsheet to access any row, column or cell. The calculate option allows you to enter labels, values or formulas into a cell or write and enter equations that will act upon the data already within the spreadsheet. You can also enter bar graphs into a cell in this option. Absolute / relative replication, down / across a column / row, is also allowed by this option. Also this option allows the automatic calculation of the entire spreadsheet with one single command. Print allows you to output to either the ZX / TS printer the entire spreadsheet by column-sets and row-pages through use of the COPY command. The entire spreadsheet may be saved on cassette tape or you may clear all data from it or erase the program from RAM entirely. The most salient advantage provided by an ESC over specifically vertical applications software is that an ESC provides a reusable framework with which you can compose any specific financial model rather than just be limited to only one statically fixed format for storing, displaying and manipulating numerical data.

\$11.95

\$1.50 SHIPPING AND HANDLING / PROGRAM

A.F.R. SOFTWARE

**- 1605 Pennsylvania Avenue, No. 204
Miami Beach, Florida 33139
(305) 531-6464**

**FL residents add 5% sales tax
Dealer inquiries invited**

PORT PROJECTS

Get the LED-OUT

By P. J. Donnelly
Long Island Sinclair Timex User Group
10 Idle Day Drive
Centerport, N.Y. 11721

In any number of monitoring and/or control uses for your ZX/TS, you may find that using a TV set to check on your system is inconvenient or impractical. This would be so, for example, in outdoor or dirty environments or when you simply wish to check the status of one or two system parameters without the bother of hooking up a TV or monitor. One fairly easy and inexpensive way to do this is to use seven-segment LED (light-emitting diode) displays as your system output. Your ZX/TS bus doesn't have enough power in reserve to directly drive these displays, but a few simple IC's and an I/O board can give you this capability.

You must have an 8-bit output port. I used Ener-Z's Report Generator board (reviewed in SWN 1:5), but JK Audio's 310 and even Byte-Back's BB-1 module, without the relays (see SWN 1:4), should be usable as a buffered output from your computer. The only other hardware you'll need, aside from miscellaneous wire and some resistors, is a 7416 hex inverting buffer, a 4511 BCD (Binary Coded Decimal)-to-7 segment decoder/driver, and a 3-character 7-segment LED display (MAN3). A schematic of this simple circuit is shown in Figure 1.

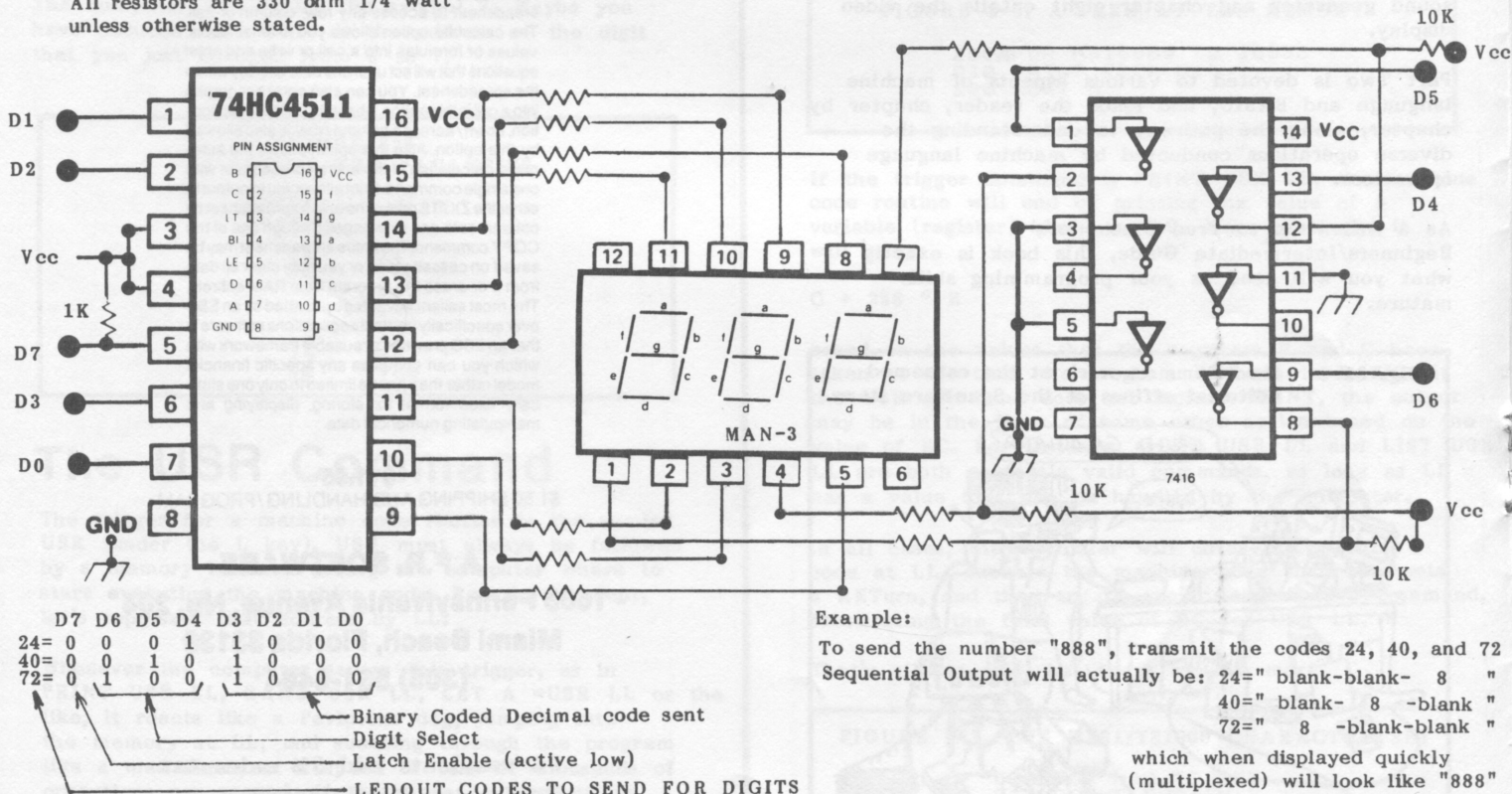
This circuit represents a "middle ground" approach to producing a 3-digit output display. More hardware (latches, etc.) can be used, resulting in permanently lighted displays, or we can use software to create a controlling outer loop to keep the monitoring function as the main operating system. The combination of hardware and software I chose represents a system which uses multiplexing to provide the appearance of a full three-digit display.

As you can see from Figure 1, we are using the ZX's data lines D0 through D3 to send out the Binary Coded Decimal value of one of three digits (units, tens or hundreds). The correct digit is determined by the data lines D4 through D6 which complete the path to ground for the appropriate display. Line D7 enables and latches the appropriate elements of the display. Multiplexing is provided by the ML software driver routine shown in Figure 2.

After data is POKed into the three buffer locations (Fig. 3a) the ML program outputs the specially constructed character code for each digit in sequence. This operation occurs so rapidly that the eye is fooled into thinking each digit is lit all the time. If you'd like to see what's really happening, you can either increase the dwell time between digits or use the BASIC listing in Fig. 4. The BASIC listing requires no machine code, and is useful in understanding how the system works, but is, of course, slow.

LEDOUT SCHEMATIC

All resistors are 330 ohm 1/4 watt unless otherwise stated.



Hex Disassembly				Decimal List				
Addr	Code	Label	Mnemonics	Comments	Address	Value	Address	Value
409E	01FF00	LDOT	LD BC,00FF	Load # of cycles	16542	1	16569	64
40A1	219740	MAIN	LD HL,HUND		16543	255	16570	13
40A4	3E04		LD A,04	# (5) of digits to send	16544	0	16571	200
40A6	F5	ENTR	PUSH AF	Save A-flags	16545	33	16572	195
40A7	7E		LD A,(HL)	Get output value from 4097h	16546	151	16573	161
40A8	328240		LD (4082),A	POKE 16514 with value	16547	64	16574	64
40AB	F1		POP AF	Get original count	16548	62	16575	201
40AC	23		INC HL	Point to next digit	16549	4	16576	17
40AD	3D		DEC A	only (A-1) digits left	16550	245	16577	0
40AE	CABA40		JP Z 40BA	Done all digits?	16551	126	16578	64
40B1	CD9329	OUTP	CALL ENRZ	Send digit out	16552	50	16579	29
40B4	CD0040		CALL DLAY	Leave it lit	16553	130	16580	202
40B7	C3A640	LOOP	JP ENTR	Send next one	16554	64	16581	202
40BA	0D		DEC C	Decrement count	16555	241	16582	64
40BB	C8		RET Z	C=0 then RET to Basic	16556	35	16583	195
40BC	C3A140		JP MAIN	If C<>0 then do it again	16557	61	16584	195
40BF	C9		RET	Can't get here	16558	202	16585	64
40C0	114000	DLAY	LD DE,0040	A simple time delay	16559	186	16586	17
40C3	1D	TIM1	DEC E	for how long we	16560	64	16587	170
40C4	CACA40		JP Z DLA2	leave each digit on	16561	205	16588	0
40C7	C3C340		JP TIM1		16562	137	16589	29
40CA	11AA00	DLA2	LD DE,00AA	This is the outer	16563	41	16590	200
40CD	1D	TIM2	DEC E	loop of the time	16564	205	16591	195
40CE	C8		RET Z	delay	16565	192	16592	205
40CF	C3CD40		JP TIM2	Keep looping	16566	64	16593	64
40D2	C9		RET		16567	195	16594	201
					16568	166		


```

1 REM 00000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
2 REM 81 CHARS ABOVE (MIN.)
10 FOR X=16542 TO 16594
15 SCROLL
20 PRINT X;
25 INPUT D
30 POKE X,D
35 PRINT "=";D
40 NEXT X
45 PRINT "DELETE LINES 2-45"

```

Note: Lines 5 to 130 initialize values in the buffer.
50 to 220 serve as a subroutine to which you would
GOSUB from your own program starting at line 1000.

The software is applicable to the Ener-Z Report Generator as it stands. However, you can change the address of the call at 16561d to the address of your own OUTPUT subroutine. The R.G. board uses a PIO and is I/O mapped. For memory-mapped systems (like the BB-1) your OUTPUT subroutine would consist of simply putting the value into the accumulator, loading the designated memory location (3FFE on the BB-1) with the value, and returning to the driver.

I built this circuit on a small ACE (all-circuit evaluator) board and obtained its power from the +5V regulator on the R.G. board. The values for the current limiting resistors to the display are not critical, but should be as large as possible to conserve power, while still giving adequate brightness. Don't overlook the 10K pull-up resistors for the 7416, or forget to tie unused inputs to ground.

```

250 REM A/D OUT
260 REM FOR ENERZ BOARD
261 REM CHOOSE CHANNEL 0
270 POKE 16514,0
275 FAST
280 LET L=USR 10602
290 REM X IS A SCALE FACTOR
300 LET X=1
310 LET D=X*PEEK 16515
320 SLOW
330 GOSUB 172
340 GOTO MENU

```

2-BIT SOFTWARE

P.O. Box 2036 Del Mar, CA 92014
Dept SWM1 (619) 481-3629

16KQUALITY IN 2K MEMORY

Games and Educational Programs
for Timex/Sinclair Computers.

FOUR GAMES ON
EVERY CASSETTE!

**FREE
CATALOG**

DEALER INQUIRIES
INVITED

TIMESCREEN™
ZX81,16K



- Creativity and planning aid
- In BASIC, with extensive use of POKE statements
- Routines to enter and rearrange data
- 3 screen formats
- Calendar
- 10 screens in 16K; 36 screens in 32K
- Use with any printer that will accept COPY routine

Cassette and manual, \$9.00 (VA res. add 4%) +\$1.00 s&h. Mail-order only, from:

HAWC™
Programming
4604 Apple Tree Drive
Alexandria, VA 22310

Figure 4. BASIC LEDOUT DRIVER FOR USE WITH ENER-Z REPORT GENERATOR

```

1 REM  AAAAAAAAAAAAAAAAAAAAAA
2 THIS IS A WORK AREA
3 FAST
4 REM BASIC LEDOUT
5 LET H=0
6 LET T=0
7 LET U=0
8 LET OUTPUT=10648
9 LET BUF=16514
10 DIM V(4)
11 REM MAIN
12 SLOW
13 PRINT "INPUT DECIMAL VALUE"
14 INPUT D
15 PRINT D
16 LET H=64+INT (D/100)
17 LET T=32+INT ((D-(H-64)*100)/10)
18 LET U=16+INT (D-(H-64)*100-(H/32)*10)
19 LET V(1)=H
20 LET V(2)=T
21 LET V(3)=U
22 LET V(4)=0
23 FOR J=1 TO 50
24 FOR I=1 TO 3
25 POKE BUF,V(4)
26 FAST
27 LET X=USR OUTPUT
28 NEXT I
29 NEXT J
30 POKE BUF,V(4)
31 LET X=USR OUTPUT
32 PRINT "AGAIN?"
33 INPUT A$
34 IF A$="Y" THEN GOTO 105
35 IF A$="D" THEN GOTO 200
36 STOP
37 REM LEDOUT
38 POKE 16514,0
39 FAST
40 LET L=USR 10602
41 LET D=PEEK 16515
42 SLOW
43 GOTO 115
44 STOP

```

The code shown would actually be a subroutine called from a main menu. The display is normally blank. By touching, say, "T" on my keyboard, I would be selecting a menu item which calls the Temperature output subroutine. All this can be done without the need of a TV screen, if I know in advance (and I do) which key to press. The system could be expanded, with more chips, lots of software and perhaps even LCD displays (for low power consumption) to actually let you program your computer without a TV. A consideration here would be to use hex buffer drivers instead of BCD for hexadecimal monitoring.

Note too, that I used "junk box" parts from around my shop to build this particular display. A proper design would use perhaps a 74LS248 for the latch driver (pinout is the same as the 4511.) Also remember NOT to output codes which will turn on all three digits at one time (e.g. binary code 01111111 produces all 8's) as the current drain will be quite high, and perhaps more than your wall-plug supply can handle.

LEDOUT CODES TO SEND FOR DIGITS IN VARIOUS COLUMNS

DIGIT in:	100's col.	10's col.	1's col.
0	64	32	16
1	65	33	17
2	66	34	18
3	67	35	19
4	68	36	20
5	69	37	21
6	70	38	22
7	71	39	23
8	72	40	24
9	73	41	25

MACHINE CODE TOPICS

How to Write Relocatable Z80 Code

Peter D. Hoffman
Delphic Enterprises
5618 Martinique Dr.
Corpus Cristi, Tx. 78411

When Delphic Enterprises decided to write some programming utility routines (renumber, string search, etc.), we made the decision that the code should be relocatable. Relocatable means that the code will execute, without modification, when placed at any available address. We found that carrying this out technically, was not a trivial problem. The techniques that had to be developed to make the code completely relocatable are the subject of this first article.

This is the first in a planned series on advanced machine code techniques on the ZX/TS computers. I will assume that you have an understanding of the Z80 machine code instruction set and some knowledge of the Sinclair specific topics such as organization of storage, system variables, character set and some acquaintance with the ROM.

I strongly recommend that you obtain the "Sinclair ZX81 ROM Disassembly" book by Dr. Ian Logan. "Machine Code programming on the Sinclair," By Toni Baker, is another good book. This is one of the most under-appreciated books in this area. If you can work through this book, including all the exercises, you will know how to program in machine code when you are finished.

I will begin by reviewing some of the instructions that are available in the Z80 set that might assist in making the programming relocatable. In particular, the instructions for transferring execution program execution to somewhere, other than the next sequential instruction, will be the key to achieving relocatable code. These are jumps and calls.

Jump Relative

The instruction that is probably most familiar, is the jump relative, JR, either conditional or unconditional. The limitation, of course, is that this jump can only be (effectively) +127 or -128 from the address of the next instruction, which is in the program counter (PC). It is possible, although awkward, to overcome this limitation by stringing several JR commands at intervals of slightly less than 128 bytes through your machine code. The result would be relocatable.

The absolute jump instruction, also conditional or unconditional, will allow the execution to be transferred within the normal 64K address space of the Z80. We can use these instructions to jump to fixed address routines, such as those within the ROM. They would not be useful in jumping to a routine that is relocatable.

Jump (HL)

There is a variation of the JP instruction that is useful, as we shall see. The JP (HL) instruction will jump to the location contained in the HL register pair. There are also analogous instructions, JP (IY) and JP (IX). If the relocatable address can be developed in the HL registers, then the JP (HL) instruction will allow us to have relocatable code.

We can develop relocatable code that will jump, by use of the JP (HL) instruction. If we have some sort of base address (similar to the index addressing mode), then all jumps could be made relative to that base address. By adding a base address and an offset in the HL register, the correct relocatable address would be available for the JP (HL). The base address is the only item that would vary, depending on where the code is located.

Let the base address be the entry point into our machine code and the same as the first byte of a block of machine code. For simplicity, this will be 2000h, although it doesn't have to be. Once the code is written, the base address could be any address that is allowed on the particular computer system.

Call Subroutine

The CALL instruction, both conditional and unconditional, has only the absolute addressed version. This instruction can be used to call fixed routines, such as those within the ROM. There is no CALL instruction that will call an address in a register pair like JP (HL).

This brings us to the major problem in the way of writing relocatable code for the Z80; CALLing your own relocatable subroutines. The Z80 opcode, CALL, allows only absolute addresses. Ideally, we would like to have an instruction, something like CALL (IY+dd), where the displacement could be more than +/- 128, but such an instruction does not exist. The next best solution is to have an instruction such as CALL (HL). The key point here is that we want the action that the CALL instruction provides; i.e., the address of the next instruction in the PC is pushed onto the stack, so that a later return will bring execution back to the same place. While the CALL (HL) instruction doesn't exist either, the JP (HL) instruction will at least get to the subroutine. The problem is then to get back when the subroutine is finished.

A solution shown below is to develop the addresses for where the "call" is to jump to (the subroutine) and the return. For the example, we assume the base address of 2000h is in the BC register pair.

LD HL, 0xxx	offset for the return address
ADD HL, BC	add offset and base address
PUSH HL	stack address for return
LD HL, 0yyy	offset for subroutine entry
ADD HL, BC	add subroutine offset and base
JP (HL)	jump to subroutine

Execution of these lines causes the computer to JUMP to a subroutine the address of which is the BASE plus the OFFSET (0yyy--stored in HL). At the conclusion of the subroutine, the RET instruction POPs the return address (BASE plus 0xxx) off the stack and jumps back to the original code.

This coding does result in relocatable but awkward code. If the coding moves relative to the base address, then all of the offsets must be changed. This occurs a lot as code is debugged. One strategem is to put thoroughly debugged subroutines out of the way at the beginning or end of the planned address space. This way there is no movement of the subroutine relative to the base address, and the offset will not change. This still leaves the problem of the changing offset for the return address.

Call to Jump (HL)

What we would really like to have, is to automatically stack the correct return address and make the JP (HL). Fortunately, there is a solution that will accomplish this objective. Residing at several places in the ROM (i.e., at fixed addresses) are JP (HL) instructions. For instance, there is a JP (HL) instruction at the address 0044h (this location is not available for those using the video upgrade noted elsewhere in this issue. Try 0ACAh for TS1000 and 1264h for 2068 machines). Thus the following piece of code will accomplish the objective of being relocatable.

LD HL, 0yyy	offset for the subroutine entry
ADD HL, BC	add offset and base address
CALL 0044	jump to subroutine

This coding will automatically stack the address following the CALL for the eventual return, and then make the jump to the subroutine via the JP (HL) instruction located at 0044h. More importantly, this coding does not have to be modified every time it moves relative to the base address, as long as the subroutine location remains fixed in offset (as previously recommended).

The BASE Address... How to Find It

On the Sinclair computer, the machine code routine is entered by a command such as, PRINT USR or RAND USR, followed by an address. This address ends up in the BC register pair, and is available on entering your machine code routine. This becomes our entry; the base address from which all addresses are referenced for relocatability. This address can be used immediately, pushed onto the stack or placed in a memory location for later retrieval. There are a number of memory locations that are relatively secure. I say relatively, since more and more programs are using identical storage locations and conflicts do develop.

...And Where to Store It

Most TS machine coders, of course, know about the "unused" locations in the system variable area. These locations are 4021h, 407Bh and 407Ch. Less well known, are several other locations that can also be used without interference with or by the 8K operating system. The sixth (mem-5) calculator register, located at 4076h to 407Ah is not used by the operating system. It apparently was a spare for future ROMs. Values can be placed there without problems. This is the area (4079h and 407Ah) that I chose to place my entry/base address for the relocatable machine code that I wrote.

Another location that can be used with impunity is 4009h, known as VERSN. This was a system variable intended to provide identification of various versions of modified or expanded ROMs. This was never utilized, and in fact, the current operating system never reads or writes to that location, other than initially setting it to zero.

The above locations can be used at any time without disturbing or being disturbed by the operating system. Furthermore, as long as you stay within your own machine code program, there are quite a few of the other variables that can also be used for temporary storage. The operating system will simply write over them later.

A problem can arise when another program, in the machine at the same time, uses the same locations. This happened when using Ray Kingsley's HOT Z program to test my own machine code program. The single stepper mode of that program uses the same mem-5 register for variables. I had to be very careful NOT to execute any instruction that loaded to mem-5 locations when single stepping or a crash would occur.

Now you may ask, why not keep the base address in the IX or IY registers rather than in a pair of system variables if there are potential conflict problems? The difficulty with this approach is that the IX and IY registers are already spoken for in the operating system. The IX register is used during the SLOW display. If your machine code never switches to SLOW mode (no pauses, input or display), then you could use the IX register. Using the IY register is more feasible. The IY register normally contains the address 4000h. This makes for compact reference to single bytes of the system variables, by use of commands that use the indexed addressing mode such as, BIT 7, (IY+d), or LD (IY+d),nn, and so forth. In my own case, since I was making extensive use of the system variables, it was useful to retain the IY register containing 4000h, in order to address those variables. If you use the IY register, it must be reset to 4000h upon exiting the machine code routine. This is most easily done by exiting through the "STACK-BC" routine at 1520h, which is what the USR function does.

Finally, let me address the problem of reading and writing data values imbedded in your code that must be rendered relocatable. I personally did not have the problem of writing to relocatable locations, since our software is epromable, and therefore you can't write to it. The same strategem applies here as for the jumps and calls, i.e., all addresses must be made relative to some base address, which necessitates writing the code to use (HL) address referencing, such as LD r, (HL).

PART 2

TMS9918A VIDEO UPGRADE

This second installment involves some explanation of the changes to the operating system and how they have been implemented. By actually placing the contents of the ROM on eprom, you can change any routine, implement extra commands outside of the 8K rom and have BASIC control of your new routines. It is only necessary to know where the table of information is, that tells the operating system

where to go to execute the BASIC command (next time). Since The VDP chip creates the display and has its own memory, a few Sinclair routines become unnecessary. These areas are precisely where the new instructions go to make our ZX/TS computers compatible with the VDP chip. To increase compatability further, some other routines are "patched" into as well.

Now the Software Side

VIDEO DISPLAY PROCESSOR INTERFACE ROM CHANGES

The VDP interface rom, mounted in board "B"'s Ea socket, is identical to the normal Timex/Sinclair rom, but with the following annotated changes:

This routine, RST38h, sends DE to the VDP's internal registers or address pointer.

```
0038 7B      R538 LD A,E      ;Get the least significant byte to send
0039 D37F    OUT (7F),A      ;Send it to the VDP
003B 7A      LD A,D      ;Get the most significant byte to send
003C D37F    OUT (7F),A      ;Send it to the VDP
003E C9      RET          ;Done-end restart
```

This is part of a patch in the FAST routine. It is called from 02ECh.

```
003F F5      DINT PUSH AF      ;Save AF
0040 D5      PUSH DE      ;Save DE
0041 11C081   LD DE,81C0      ;Set DE to send VDP 81C0h after next interrupt
0044 76      HALT          ;Wait until after next interrupt, so we can complete uninterrupted
0045 FF      RST 38H        ;Send 81C0h to VDP, to disable future interrupts.
0046 D1      POP DE        ;Restore DE
0047 F1      POP AF        ;Restore AF
0048 C9      RET          ;Return to 02EFh.
```

This routine is the new non-maskable interrupt handler. It will:

1. Copy the display file every fourth time (15 times per second).
2. Scan the keyboard with debounce.

```
0066 08      NMIIH EX AF,AF"   ;Get the counter
0067 3C      INC A           ;Increment the count
0068 2005     JR NZ,KSCN      ;Jump over display copy until A=00
006A CD7702   CALL CPDF       ;Send the display file to the VDP
006D 3EFC     LD A,FC         ;Set the counter to skip sending it for the next three interrupts
006F CD1702   KSCN CALL SCAN   ;Scan the keyboard
0072 08      EX AF,AF"       ;Save the counter for the next time
0073 F5      PUSH AF         ;Save AF
0074 DB7F     IN A,(7F)       ;Clear the VDP's interrupt output
0076 F1      POP AF          ;Restore AF
0077 ED45     RETN           ;Return from the non-maskable interrupt.
0079 00      NOP             ;Five bytes open.
007A 00      NOP
007B 00      NOP
007C 00      NOP
007D 00      NOP
```

This is a patch in the SAVE/LOAD end check routine, for border color changes during SAVEing and LOADING.

```
0204 C3AE02   JP CHBD        ;Jump to patch at 02AEh, for border color update.
```

8K ROM Changes Cont.

This routine enables VDP interrupts for SLOW mode operation and scans the keyboard one time.

```

020F CBFE      EINT SET 7, (HL)      ;Set bit 7 of CDFLAG for SLOW mode operation
0211 D5        PUSH DE              ;Save DE
0212 11E081    LD DE, 81E0h         ;Set DE to send 81E0h to VDP
0215 FF        RST 38H              ;Send 81E0 to VDP to enable its interrupt generator
0216 D1        POP DE               ;Restore DE.
0217 F5        SCAN PUSH AF         ;Save AF
0218 C5        PUSH BC              ;Save BC
0219 D5        PUSH DE              ;Save DE
021A E5        PUSH HL              ;Save HL
021B 180C      JR 0229              ;Go and scan the keyboard.

```

This routine is part of the FAST mode keyboard scan loop routine.

```

021D CD7702    FSLP CALL CPDF        ;Send the display file to the VDP
0220 010E4     LD BC, E406           ;Initialize BC for the delay loop
0223 10FE      LOOP DJNZ LOOP        ;Delay loop
0225 0D        DEC C                 ;used to get accurate
0226 20FB      JR NZ LOOP            ;PAUSE.

```

This is a patch in the keyboard debounce routine.

```

0253 213B40    LD HL, CDFF          ;Point HL at CDFLAG
0256 CB86      RES 0, (HL)           ;Reset bit 0, signifying no valid keypress
0258 2005      JR NZ CONT            ;Jump if no valid key depression was found
025A CB7E      BIT 7, (HL)           ;Are we in FAST or TEMP FAST mode?
025C C8C6      SET 0, (HL)           ;Set bit 0 of CDFLAG. A valid keypress has been found.
025E C8        RET Z                 ;Return if we are in FAST or TEMPFast mode.
025F 212740    CONT LD HL, DBNCE     ;Point HL at the system variable DBNCE
0262 7B        LD A, E               ;A = old key position
0263 FEFE      CP FE                 ;Set carry, if valid keypress last time through
0266 9F        SBC A, A              ;A = 00 if no valid keypress, or A = FF if there was last time
0268 061F      LD B, 1F              ;Initialize B for ANDing with A
026A B6        OR (HL)               ;A = (DBNCE) if no valid key, or A = FF if there was (to init.)
026B A0        AND B                 ;A = (DBNCE) if no valid keypress, or A = 1F if a valid one
026D 1F        RRA                   ;A = 0F if keypress or A = 07, 03, 01, then 00 for no key (4)
026E 77        LD (HL), A            ;Store in (DBNCE) for next time (need 4 nokeys now if valid)
026F FDCB3B7E  BIT 7, (CDFF)         ;Are we in FAST or TEMPFast?
0270 28AB      JR Z FSLP             ;Loop again if we are
0272 E1        POP HL               ;Restore HL
0273 D1        POP DE               ;Restore DE
0274 C1        POP BC               ;Restore BC
0275 F1        POP AF               ;Restore AF
0276 C9        RET                  ;Return to 0072h. We are in SLOW mode.

```

This routine sends the entire DISPLAY FILE to the VDP memory.

```

0277 C5        CPDF PUSH BC          ;Save BC
0278 D5        PUSH DE              ;Save DE
0279 E5        PUSH HL              ;Save HL
027A 110044    LD DE, 4400h         ;Set DE for RST38h
027D FF        RST 38H              ;Set VDP address pointer to the beginning of its name table.
027E 1618      LD D, 18              ;Set D to send 24 lines of characters
0280 3E76      LD A, 76              ;Set A to 76h for ENTER token comparisons
0282 2A0C40    LD HL, (DFIL)         ;Point HL at the DISPLAY FILE
0285 23        INC HL               ;Skip the ENTER token
0286 013F20    LD BC, 203F          ;Set BC to 20h (32 decimal) characters to go out port 3F
0289 BE        LP-2 CP (HL)          ;Is this character the ENTER token?
028A 230B      JR Z PAD              ;Jump to PAD out this line if it is
028C EDA3      OUTI                  ;Send one character, point to the next one and decrement count
028E 20F9      JR NZ LP-2            ;Loop until 32 characters are sent
0290 15        DCLN DEC D            ;Decrement line counter
0291 20F2      JR NZ LP-1            ;Loop until 24 lines are sent
0293 E1        POP HL               ;Restore HL
0294 D1        POP DE               ;Restore DE
0295 C1        POP BC               ;Restore BC
0296 C9        RET                  ;Done sending DISPLAY FILE, so RETURN from call
0297 D33F      PAD OUT (3F), A        ;Send an ENTER token to the VDP
0299 10FC      DJNZ PAD              ;Loop until the entire line is padded out
029B 18F3      JR DCLN              ;Jump to decrement line count

```

GOOD NEWS! Hal Tronics, Inc., PO Box 1101, Southgate, MI 48195, has the TMS9918A chip for \$9.95. That reduces the cost of the Video Upgrade by \$30.00. Also, using 5V 4116 RAM chips eliminates the need for the extra power supply, reducing the cost of this system even more. Get details from John Olliger.

BAD NEWS! Don't buy your crystal from Active Elec. Their crystal is not the right type for this interface. If you have a KDS crystal, then it may be necessary to put a trim cap across pins 39 and 40 of the VDP in order to "lock in" the color on some sets. Try 5 to 60 picofarads.

8K ROM Changes Cont.

This routine is part of a patch in the Timex/Sinclair printer driver routine. It allows access to the dot patterns in the VDP video ram, for delivery to the 2040 printer.

```

029D D5      PPCH  PUSH DE      ;Save DE
029E 11000A   LD DE,0A00      ;Set DE for base address of the normal character dot patterns
02A1 3003     JR NC,CALC      ;Jump if no carry. Code is for a normal character.
02A3 11000E   LD DE,0E00      ;Set DE for base address of inverse code dot patterns
02A5 19      CALC  ADD HL,DE   ;Form correct video ram address by adding the displacement
02A7 EB      EX DE,HL         ;DE = video ram address of dot patterns
02A8 FF      RST 38H          ;Set the VDP address pointer
02A9 D1      POP DE           ;Restore previous DE
02AA DB3F     IN A,(3F)        ;Get the dot pattern from VDP ram
02AC 4F      LD C,A           ;Store it in C
02AD C9      RET              ;Done, RETURN

```

This routine is a patch in the LOAD/SAVE endcheck routine. It updates the screen border colors (similar to a 2068) during a LOAD or SAVE. It is entered from 0204h. This routine is necessary, because the bars are no longer available with the new video system. This will give some indication of SAVE/LOAD in progress.

```

02AE EB      CHBD  EX DE,HL    ;Return next bytes location in HL for patch
02AF 5D      LD E,L           ;Least significant byte of address will be the border color
02B0 1687    LD D,87          ;Set D so RST38h will load VDP register 7
02B2 FF      RST 38H          ;Send new border color to VDP
02B3 D0      RET NC           ;Return to SAVE/LOAD another byte if no carry. Not done yet.
02B4 CD081E  CALL LDRC        ;Reset the VDP registers to their power up default values
02B7 E1      POP HL           ;Restore HL for patch
02B8 C30702  JP SLO?          ;Jump to SLO? for patch

```

This is a shortening and very slight modification to the keyboard scan routine.

```

02D6 A7      AND A            ;Clear carry flag
02DC C9      RET              ;Done, so return from call

```

This routine gets one byte from the normal rom character generator area. HL points to the byte desired and the fetched byte is returned in A.

```

02DD DBFD    GETB  IN A,(FD)    ;Enable rom character generator area
02DF 7E      LD A,(HL)         ;Get the data from the rom
02E0 D3FD    OUT (FD),A        ;Disable the rom character generator and enable entire eeprom
02E2 C9      RET              ;Return from call
02E3 00      NOP               ;4 bytes open
02E4 00      NOP
02E5 00      NOP
02E6 00      NOP

```

This is a patch in the TEMP FAST mode routine.

```

02EC CD3F00  CALL DINT         ;Call disable VDP interrupts routine

```

This is a patch in the initialization routine, to allow initialization of the VDP and its memory.

```

0410 CD691E  CALL INIT         ;Initialize the VDP

```

This is a patch in the Timex/Sinclair printer driver routine, to allow access to the VDP pattern generator area, so that any character on the video screen can be sent to the 2040 printer without any additional software.

```

08B0 CD9D02  CALL PPCH         ;Call the printer driver patch routine.

```

MORE GOOD NEWS! To run the Memotech I/F with this system requires no extra decoding. A very simple software modification will do the trick!

8K ROM Changes Cont.

All the rest of the new software resides in the 1E00 to 1EFF block of the eeprom, opposite where the character dot patterns are stored in the normal rom. This, and the bank select logic used in its implementation, was done so that no additional memory space is occupied by this project.

The Following routine initializes the VDP's internal registers

```

1E00 00      DATA
1E01 C0      DATA
1E02 01      DATA
1E03 40      DATA
1E04 05      DATA
1E05 04      DATA
1E06 00      DATA
1E07 F1      DATA

1E08 21001E  LD RG LD HL,DATA ;Point HL at the data to send
1E09 1680    LD D,80 ;Initialize D to 80h to tell the VDP which register to load
1E0A 0608    LD B,08 ;Set loop counter for 8 pieces of data
1E0B 5E      LD E,(HL) ;Get byte of data to send
1E0C FF      RST 38H ;Send data byte, then register destination byte
1E0D 14      INC D ;Increment register destination pointer
1E0E 23      INC HL ;Point at next piece of data
1E0F 10FA    DJNZ LP-3 ;Repeat until all 8 registers are loaded
1E10 09      RET ;Done, RETURN

```

This routine sends the normal character generator dot patterns to the VDP ram pattern generator area

```

1E16 110058  SGNR LD DE,6800 ;Set DE so RST38h will point the VDP address pointer at 2800h.
1E17 FF      RST 38H ;Set the VDP address pointer
1E18 21001E  LD HL,DATA ;Point HL at the start of the roms character generator
1E19 010200  LD BC,0002 ;Set byte counter for 512d bytes
1E1A CDD002  LP-4 CALL GETB ;Get a byte from the roms character generator area
1E1B D33F    OUT (3F),A ;Send it to the VDP ram
1E1C 23      INC HL ;Point at next byte in rom
1E1D 10F8    DJNZ LP-4 ;Repeat until 256d bytes have been sent
1E1E 00      DEC C ;Decrement block counter
1E1F 20F5    JR NZ LP-4 ;Loop to send a total of 512d bytes
1E20 CD501E  CALL SND0 ;Send 512d 00's to VDP, for space character if bit 6 is set in code
1E21 21001E  LD HL,DATA ;Point at beginning of the rom character generator again
1E22 0E02    LD C,02 ;Set block counter for 2 blocks of 256d bytes
1E23 CDD002  LP-5 CALL GETB ;Get a byte from the rom character generator
1E24 2F      CPL ;Compliment the data for inverse characters
1E25 D33F    OUT (3F),A ;Send it to the VDP ram
1E26 23      INC HL ;Point at next byte
1E27 10F7    DJNZ LP-5 ;Repeat 256 times
1E28 00      DEC C ;Decrement block counter
1E29 20F4    JR NZ LP-5 ;Loop to send 512d bytes
1E2A 180F    JR SND0 ;Jump to send an additional 512d bytes of 00's

```

The next routine initializes the VDP color table for white on black characters.

```

1E41 110050  COLR LD DE,5000 ;Set DE to point the VDP's address pointer to 1000h
1E42 FF      RST 38H ;Set the VDP's address pointer
1E43 3EF1    LD A,F1 ;Set A for white characters on black background
1E44 010120  COLA LD BC,2001 ;Set byte counter to send 32d bytes
1E45 1808    JR SND0 ;Jump to send it to VDP's ram

```

The following routine clears the VDP ram's sprite attribute area of garbage.

```

1E4C 110042  CLSP LD DE,4200 ;Set DE to point the VDP's address pointer at 0200h
1E4D FF      RST 38H ;Set the VDP's address pointer
1E4E AF      XOR A ;Let A=00
1E4F 010200  SND0 LD BC,0002 ;Set byte counter for 512d bytes to send
1E50 D33F    OUT (3F),A ;Send A to VDP ram
1E51 10FC    DJNZ SND0 ;Repeat B times
1E52 00      DEC C ;Decrement block counter
1E53 20F9    JR NZ SND0 ;Repeat a total of ((C-1)*256)+B times
1E54 C9      RET ;Done, so RETURN from call

```

This routine sends (HL) to port (C), decrements B and repeats until B = 00.

```

1E5C EDA3    SEND OUTI ;Send (HL) to (C), dec B and set Zero flag if B = 00
1E5D 13      INC DE ;Increment address counter
1E5E 20FB    JR NZ SEND ;Repeat until B = 00

```


8K ROM Changes Cont.

This routine adds 8 to DE.

```

1E61 E5      .SKIP PUSH HL      ;Save HL
1E62 210800  LD HL,0008        ;Set HL for addition
1E65 19      ADD HL,DE         ;HL=HL+DE
1E66 EB      EX DE,HL         ;DE=DE+8
1E67 E1      POP HL           ;Restore HL
1E68 C9      RET              ;Done, so return to caller

```

This routine totally initializes the VDP. It is called from 0410 on power up.

```

1E69 CD081E  INIT CALL LDRG      ;Initialize the VDP's internal registers
1E6C CD161E  CALL SNGN        ;Send rom character generator to VDP pattern generator
1E6F CD411E  CALL COLR        ;Set the VDP ram's color table
1E72 CD4C1E  CALL CLSP        ;Clear the VDP ram's sprite attribute table
1E75 08      EX AF,AF"        ;Get screen copy counter
1E76 3EFC    LD A,FC          ;Initialize it to FCh
1E78 08      EX AF,AF"        ;Put it back
1E79 C39A14  JP CLER          ;Jump to 149A for patch

```

This routine (USR 7804) sends lower case and special character dot patterns to the VDP.

```

1E7C CDE702  LOWR CALL TFAS      ;Enter TEMP FAST mode
1E7F FDC83BCE SET 1,(CDFG)        ;Set flag showing that we are in lower case mode
1E83 11586C  LD DE,6C58        ;Set DE to point to the VDP address pointer at 2C58h
1E86 FF      RST 38H          ;Set the VDP address pointer
1E87 21E01E  LD HL,1EE0        ;Point HL at lower case special character patterns
1E8A 013F10  LD BC,103F        ;Set B for 16 bytes and C to port 3F
1E8D CD5C1E  CALL SEND        ;Send patterns for "@" and "#"
1E90 FF      RST 38H          ;Point VDP address pointer 8 bytes further along
1E91 0620    LD B,20          ;Set B to send 32d bytes this time
1E93 CD5C1E  CALL SEND        ;Send patterns for " ", "%", " " and " "
1E96 CD611E  CALL SKIP        ;Skip a total of two character pattern areas
1E99 FF      RST 38H          ;Point VDP address pointer 16 bytes further along
1E9A 0608    LD B,08          ;Set B to send only 8 bytes this time
1E9C CD5C1E  CALL SEND        ;Send pattern for "&"
1E9F FF      RST 38H          ;Point VDP address pointer 8 bytes further along
1EA0 0608    LD B,08          ;Set B to send 8 bytes
1EA2 CD5C1E  CALL SEND        ;Send pattern for " "
1EA5 CD611E  CALL SKIP        ;Skip a total of two character positions this time
1EA8 FF      RST 38H          ;Point VDP address pointer 16 bytes further on
1EA9 0610    LD B,10          ;Set B to send 16 bytes, (" " and " ")
1EAB EDA3    LP-6 OUTI        ;Send a byte, increment HL, decrement B, and set Z flag if B=00
1EAD 13      INC DE           ;Increment VDP address pointer
1EAE 20FB    JR NZ,LP-6       ;Repeat until B = 00 (16d times)
1EB0 06D0    LD B,D0          ;Set B to 176d to send the lower case letters
1EB2 11306D  LD DE,6D30        ;Set DE to set VDP address pointer to 2D30h
1EB5 FF      RST 38H          ;Point VDP address pointer to inverse letters area
1EB6 CD5C1E  CALL SEND        ;Send lower case letter patterns
1EB9 C30702  JP SLO?          ;RETurn via the SLO? routine

```

This routine returns the lower case letters back to their normal inverse characters. This routine is called from BASIC with a RAND USR 7868.

```

1EBF FDC83BCE CALL TFAS      ;Enter TEMP FAST mode
1EC3 CD161E  RES 1,(CDFG)      ;Reset lower case enable flag
1EC6 C30702  CALL SNGN        ;Reinitialize the VDP pattern generator area
1EC6 C30702  JP SLO?          ;RETurn via the SLO? routine

```

This last routine sets the border color and the character background and foreground colors to the value POKEd into 16424h. POKE 16424 with the desired paper/ink colors and RAND USR 7881. 16424 (MARGIN) is now a free system variable.

```

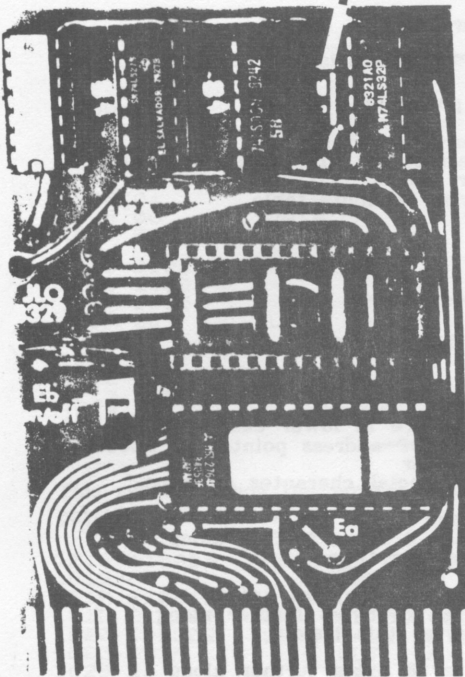
1EC9 CDE702  CALL TFAS      ;Enter TEMP FAST mode
1ECC 110050  LD DE,5000        ;Set DE to set address pointer to 1000h
1ECF FF      RST 38H          ;Set the VDP address pointer
1ED0 3A2840  LD A,(MARG)      ;Get colors to send
1ED3 CD471E  CALL COLA        ;Set the screen paper and ink colors
1ED6 5F      LD E,A           ;Let E = A
1ED7 1687    LD D,87          ;Get set to send 4028h to VDP register 7
1ED9 FF      RST 38H          ;Load VDP register 7
1EDA C30702  JP SLO?          ;RETurn via SLO? routine
1EDD 00      NOP              ;Three bytes open
1EDE 00      NOP
1EDF 00      NOP

```

YOUR VIDEO BOARDS

Photos by LOU TOTH

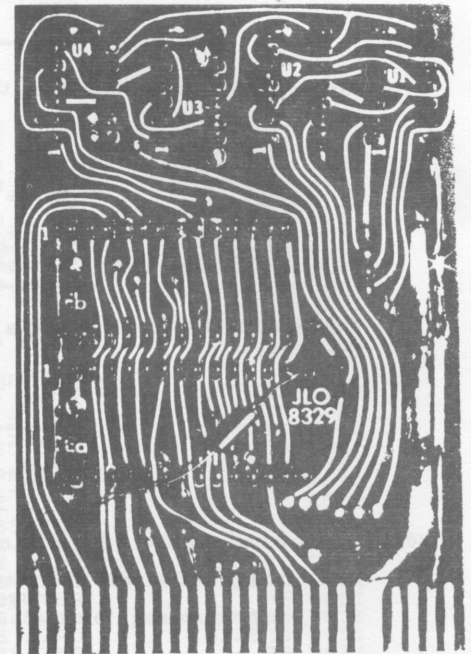
BANK SWITCHING LOGIC



This bypass cap fits nicely here.

Eb contains any other utilities that you may want to have on board. It is switchable.

Ea contains your new operating system.



The following table is the data making up the dot patterns for the lower case and special characters:

1EE0	00	3C	4E	4A	4E	40	3C	00	"a"
1EE8	00	24	7E	24	24	7E	24	00	"b"
1EF0	00	32	4C	00	00	00	00	00	"c"
1EF8	00	62	64	08	10	26	46	00	"d"
1F00	00	0C	10	70	70	10	0C	00	"e"
1F08	00	60	10	1C	1C	10	60	00	"f"
1F10	00	18	24	18	2A	44	3A	00	"g"
1F18	00	00	00	00	00	00	00	FF	"h"
1F20	00	20	10	00	00	00	00	00	"i"
1F28	20	20	20	40	00	00	00	00	"j"
1F30	00	00	38	04	3C	44	3A	00	"k"
1F38	00	40	40	78	44	44	78	00	"l"
1F40	00	00	38	44	40	40	38	00	"m"
1F48	00	04	04	3C	44	44	3C	00	"n"
1F50	00	00	38	44	7C	40	38	00	"o"
1F58	00	1C	20	78	20	20	20	00	"p"
1F60	00	00	38	44	44	3C	04	38	"q"
1F68	00	40	40	78	44	44	44	00	"r"
1F70	00	10	00	30	10	10	38	00	"s"
1F78	00	08	00	18	08	08	48	30	"t"
1F80	00	40	44	48	70	48	44	00	"u"
1E88	00	30	10	10	10	10	38	00	"v"
1F90	00	00	24	5A	5A	5A	42	00	"w"
1F98	00	00	4C	32	22	22	22	00	"x"
1FA0	00	00	38	44	44	44	38	00	"y"
1FA8	00	00	78	44	44	78	40	40	"z"
1FB0	00	00	3C	44	44	3C	04	04	"A"
1FB8	00	00	4C	32	20	20	20	00	"B"
1FC0	00	00	3C	40	3C	02	3C	00	"C"
1FC8	00	20	78	20	20	24	18	00	"D"
1FD0	00	00	48	48	48	48	34	00	"E"
1FD8	00	00	44	44	44	28	10	00	"F"
1FE0	00	00	42	42	5A	3C	24	00	"G"
1FE8	00	00	44	28	10	28	44	00	"H"
1FF0	00	00	42	22	14	08	08	30	"I"
1FF8	00	00	7C	08	10	20	7C	00	"J"

You may change any of these patterns to suit your personal preferences.

DLS BUYERS GUIDE TO SINCALIR - TIMEX PRODUCTS & SERVICES

FEATURING
PRODUCT DESCRIPTIONS
SUPPLIER INFORMATION
CROSS REFERENCES
CURRENT PRICES
NEWSLETTERS

SOFTWARE
HARDWARE
SUPPLIES
SERVICES
PUBLISHERS

FOR

ZX-80 ZX-81 SPECTRUM QL
TS-1000 TS-1500 TS-2068

We have attempted to contact over 1000 possible suppliers of products and services for these computers. The data we received is arranged so that it may easily be updated. The indexes are used to enable you to find and compare possible product choices. Comprehensive category list is invaluable in your search of a product. Price: \$20.00 (US) postpaid includes the update that's scheduled for the first quarter of 1985. Check Money Order, VISA or MASTERCARD. Sorry no COD. Stamped addressed legal envelope for complete details and a catalogue of EDUCATIONAL, BUSINESS and PERSONAL software plus our new hardware and supplies section.

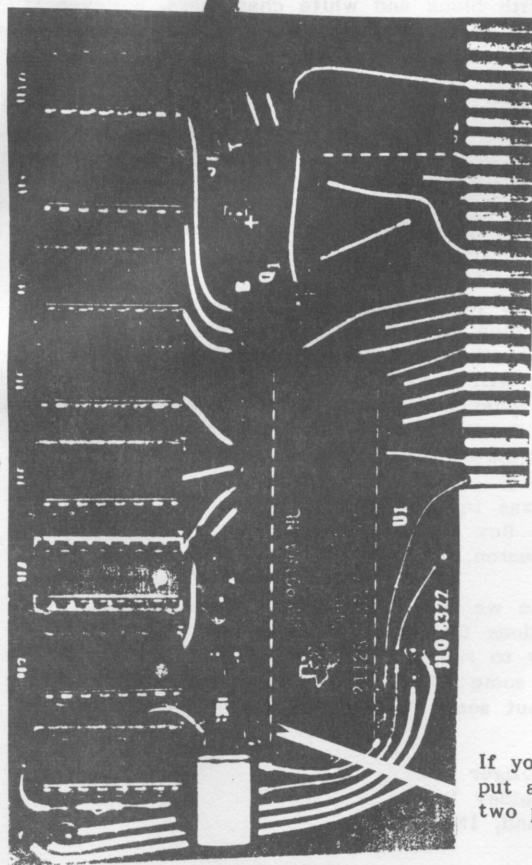
SPECIAL Silver Reed EXP-400 Daisy wheel printer parallel, friction feed, 12 CPS \$349.00 PP(US)

D LIPINSKI SOFTWARE
2737 Susquehanna Road
Roslyn, PA. 19001 USA

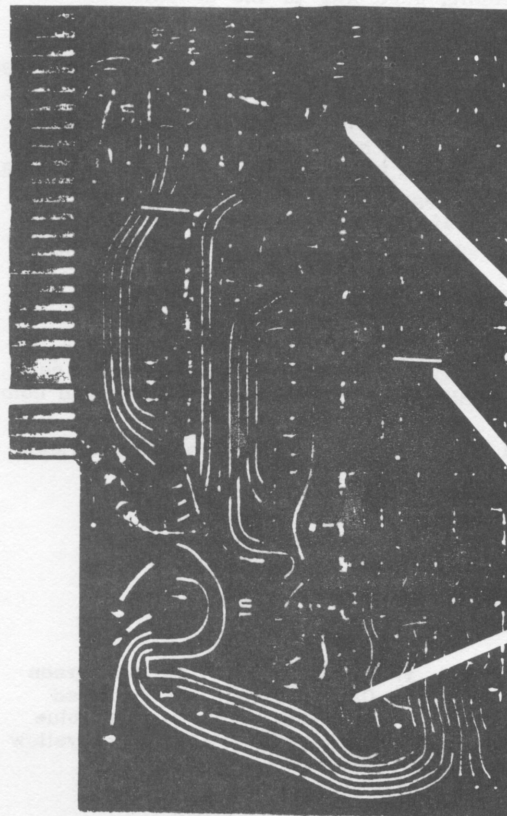
75 ohm composite
video out

We hope that your boards look like these--or better!!

16K
VIDEO
RAM



If you have a KDS crystal,
put a trim cap across these
two pins, to lock color.



watch
the
polarity
on
these
bypass
caps.

I assume at this point that your new video system is working properly without any additional memory. Note that in FAST mode the screen does not blink when you press a key. The display is continuously being generated by the VDP, and is only changed (updated) during a PAUSE or while waiting for keyboard input. On entering a BASIC line, notice that it just appears in the listing. Because the 9918A is a video display PROCESSOR, it is technically a "co-processor" to the Z80 CPU in your Timex/Sinclair computer. It frees the Z80 from the chore of creating the display by itself, therefore the computer will run much faster in SLOW-almost as fast as FAST.

Inverse , is now a ' Inverse £ is now a #
Inverse ; is now a \ Inverse - is now a -
Inverse ? is now a % Inverse : is now a ~
Inverse " is now a @ Inverse = is now a &
Inverse) is now a } Inverse (is now a (

How do you like them. Even the 2068 is missing some of these characters.

What You See is...

Now it's time to play around with your system to see just what new changes have come along. Let's put some lower case letters on the screen. Enter RAND USR 7804/ENTER. The computer will respond with its' report code. Press ENTER again and a lower case "k" will appear on the screen in the cursor position. This is the command cursor replacing the inverse K. Enter the graphics mode (shift 9) and a lower case g will be the new cursor. Now you are in the lower case mode, and can print lower case to the screen and printer. Try it. Some special characters have also been implemented to improve the output to your printer. The following INVERSE characters are redefined:

Back to Sinclairese

RAND USR 7868 will return the normal inverse characters to the screen. One problem though, since you are swapping complete character sets, you can't have mixed modes on the screen at the same time. See the listing to get more details on this. When sending text to the printer, there is no reason why modes can't be changed for a character or two and then changed back. No mixed mode is allowed on the screen only. On the same token, no pun intended, using some of the new subroutines annotated herein, there is no reason why you can't define a character set of your own to display or print.

If you build my parallel printer port (published originally in SQ#1 & 2), then a second eprom mounted in Eb can be used to contain my ASCII conversion table and driver at 2000h. Just include my printer patch at the prompt on the programming cassette. The lower case and special characters will LPRINT properly regardless of the mode that the screen is in, simply because they are all ASCII characters contained in the conversion table.

Check Out that COLOR!!

RAND USR 7881 will change the character/background (ink/paper) colors, according to the contents of 16424 (4028h). Location 16424 was the system variable MARGIN. This is used by the computer to determine whether to produce a PAL (English TV @ 50 frames/sec. and 625 line scan) or NTSC (that's us-60 frames/sec, 525 lines). This system variable now belongs to us to color our screen as we see fit. For example; POKE 16424,31 and ENTER. Now RAND USR 7881. This should bring up a familiar black on white screen. POKE 16424,241/ENTER and RAND USR 7881 will bring up the recommended white on black screen.

You can change screen colors to any of 16 colors (ink and paper) by POKEing 16424 as follows:

The 4 most significant bits will be the character color,
The 4 least significant bits will be the background color.

POKE 16424,X where X is derived from the formula
 $X=A*16+B$
A = the desired character color
B = the desired background color
RAND USR 7881

The following table lists the various colors available for you to "color" with:

0= Transparent	1= Black	2= Med. green
3= Light green	4= Dark blue	8= Med. red
6= Dark red	7= Cyan	5= Light blue
9= Light red	10= Dark yellow	11= Light yellow
12= Dark green	13= Magenta	14= Grey
15= White		

Thus, for a screen with light yellow characters on a dark blue background, POKE 16424,180. The reason? $11*16+4=180$. Black and white monitors will be clearer with black and white characters, however if you have a color monitor, you may find another color combination that you prefer.

During LOAD/SAVE operations, you will find that the bars are gone. The screen border now changes (similar to the 2068) colors rapidly, furthermore it changes ONLY when a byte has been LOADED or SAVED. If you key in LOAD "ZX"/ENTER, the border will not change color until "ZX" has been found and is being LOADED.

If you care to go further with this system and do some machine code programming to make use of the sprites, or other features of this chip, the first thing you will need is a copy of "The TMS9918A Video Display Processor Manual." Book #MP010 or MP010A are available from Texas Instruments, either through a local distributor (maybe for free), or write to:

Texas Instruments
PO Box 1443
Houston, Tx. 77001

Next time we will cover some of the optional modifications that are available to give us the capability to run machine code from anywhere in memory, some ZX80 modifications and talk a little more about some features of the VDP chip.

John I. Olinger
11601 Whitey Drive
Cumberland, IN 46229

SyncWare News

CONTENTS

For Your Support.....	2
Forum.....	3
Decimal to Hex and Back.....	3
Woods	
SAVE/LOAD Utility.....	4
Shaughnessy	
Run Memotext in RAM!!!.....	6
Nachbaur	
Basil's Compendium, PEEK POKE USR.....	9
Wentworth	
TS Intermediate/Advanced Guide.....	11
REVIEW by Marsh	
Port Projects/Get the LED Out.....	12
Donnelly	
How to Write Relocatable MC.....	15
Hoffman	
A Video Upgrade: Part II.....	16
Olinger	

Technical Director:
Fred Nachbaur
902 Hoover Street
Nelson, BC CANADA V1L-4X6

Editor:
Tom Bent
9016 Flicker Place
Columbia, MD 21045

Published by
Thomas B. Woods
P.O. Box 64,
Jefferson, NH 03583

FIRST-CLASS MAIL
U.S. POSTAGE
PAID
Jefferson, NH
Permit No. 1

Submissions, announcements, letters to the editor, and all other material of editorial interest should be sent to the Maryland address.

Copyright 1984 SyncWare News

Subscribe !

\$ 16.95

1 year (6 issues)

Add \$3 a year in Canada; all other foreign add \$5 per year.